二叉树

TEACHING COURSEWARE POWPOINT

授课时间：2025.08.18

目录

01

知识回顾

TEACHING
COURSEWARE

TEACH

知识回归

顶点 / 节点	树的基本组成单元
边	连接两个节点的线，树中边数 = 节点数 - 1，且每条边都是 “无向” 的
叶节点	无根树：度数为 1 的节点（只连一条边，是树的 “末端”）； 有根树：没有子节点的节点（从根出发的 “最后一层” 节点）。
根节点	仅在 “有根树” 中存在，是人为指定的 “起点”（用于区分父子关系）。
父 / 子节点	仅在有根树中存在：从根到某节点的路径上，前一个节点是 “父节点”，当前节点是 “子节点”
子树	以某节点为起点，包含其所有后代节点（子节点、孙节点等）和连接边形成的 “小树”（是原树的子集）。
深度 / 高度	- 深度（有根树）：根节点到当前节点的路径长度（根节点深度通常设为 0 或 1，需看题目约定）； - 高度（有根树）：当前节点到最远叶节点的路径长度（叶节点高度为 0）。
度	无根树：和图相同 有根树：节点的子节点个数

知识回归

深度	根节点到当前节点的路径长度
高度	当前节点到最远叶节点的路径长度
子树	以某节点为起点，包含其所有后代节点（子节点、孙节点等）和连接边形成的 “小树”（是原树的子集）。

02

分类

TEACHING
COURSEWARE

TEACH

分类

每个节点最多有两个子节点（子树）的树结构
子节点严格区分左子节点（Left Child）和右子节点（Right Child）
次序不能颠倒。
基本形态：
空二叉树
只有根节点
只有根节点和左子树
只有根节点和右子树
有根节点、左子树和右子树

1.叶子节点与度为 2 的节点关系：任意非空二叉树中，叶子节点数 = 度为 2 的节点数 + 1（“度” 指节点的子节点数量，叶子节点度为 0）。

分类

叶子节点数 = 度为2的节点数 + 1
步骤1：定义变量
设二叉树中：
n_0 = 叶子节点数（度为0的节点，没有子节点）
n_1 = 度为1的节点数（只有一个子节点）
n_2 = 度为2的节点数（有两个子节点）
总节点数 $N = n_0 + n_1 + n_2$
步骤2：计算边数 二叉树中，边数 = 总节点数 - 1。
步骤3：从节点度的角度计算边数 另一种计算边数的方式：
度为0的节点（叶子）贡献 $0 \times n_0 = 0$ 条边
度为1的节点贡献 $1 \times n_1 = n_1$ 条边
度为2的节点贡献 $2 \times n_2 = 2n_2$ 条边
因此，总边数 $= n_1 + 2n_2$ 。
步骤4：联立方程推导关系 根据步骤2和步骤3的边数相等：
$N - 1 = n_1 + 2n_2$
又因为总节点数 $N = n_0 + n_1 + n_2$
代入上式：$(n_0 + n_1 + n_2) - 1 = n_1 + 2n_2$
化简后：$n_0 - 1 = n_2$ 即 $n_0 = n_2 + 1$。
结论 任意非空二叉树中，叶子节点数（n_0 ）一定比度为2的节点数（n_2 ）多1。
这个性质与二叉树的形态无关，无论是满二叉树、完全二叉树还是任意形态的二叉树，均成立。

分类

定义
完全二叉树：
对于深度为 h 的二叉树，若前 h-1 层的节点数达到最大值
且第 h 层的节点从左至右连续排列，无空缺，则称为完全二叉树。

特点：叶子节点只能出现在最下层和次下层，且最下层的叶子节点集中在左侧。
基本性质
节点数范围：
深度为 h 的完全二叉树
节点数 n 满足：$2^{(h-1)} \leqslant n \leqslant 2^h-1$。

分类

重要性质：
第 i 层最大节点数： 2^(i-1)（i ≥ 1）

深度为 k 的二叉树最大总节点数： 2^k- 1（k ≥ 1）（满二叉树）

特殊二叉树：
满二叉树：深度为 k 的二叉树有 2^k- 1 个节点。每一层都填满 节点。
完全二叉树 ：深度为 k，有 n 个节点。除了最后一层，其他 层都填满。最后一层的节点都连续集中在最左边。

分类

完全二叉树的数组表示法
存储原理 用一维数组按层序遍历顺序存储完全二叉树的节点，下标从1 开始。
若节点 i 存在左子节点，则其左子节点存储在下标 2i 处；
右子节点存储在下标 2i+1 处。
父节点 i 的父节点下标为⌊i/2⌋（i>1 时）。

概念

一棵有 n 个结点的完全二叉树用数组进行存储与表示，已知根结点存储在数组的第 1 个位置。若存储在数组第 9 个位置的结点存在兄弟结点和两个子结点，则它的兄弟结点和右子结点的位置分别是（ ）。

- A. 8、18
- B. 10、18
- C. 8、19
- D. 10、19

概念

独根树的高度为 1。具有 61 个结点的完全二叉树的高度为（ ）。

- A. 7
- B. 8
- C. 5
- D. 6

概念

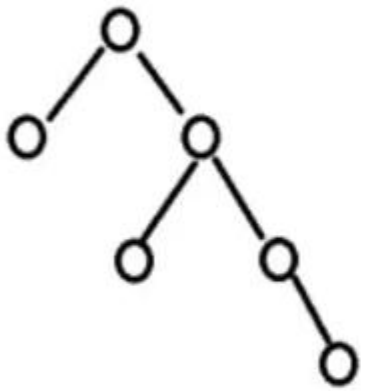
如果一棵二叉树只有根结点，那么这棵二叉树高度为 1。请问高度为 5 的完全二叉树有 （ ）种不同的形态?

- A. 16
- B. 15
- C. 17
- D. 32

概念

第 8 题

一棵二叉树如右图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点（根结点的下标为 1，若某结点的下标为 i，则其左孩子位于下标 $2i$ 处、右孩子位于下标 $2i+1$ 处），则该数组的最大下标至少为（）。



- A. 6
- B. 10
- C. 15
- D. 12

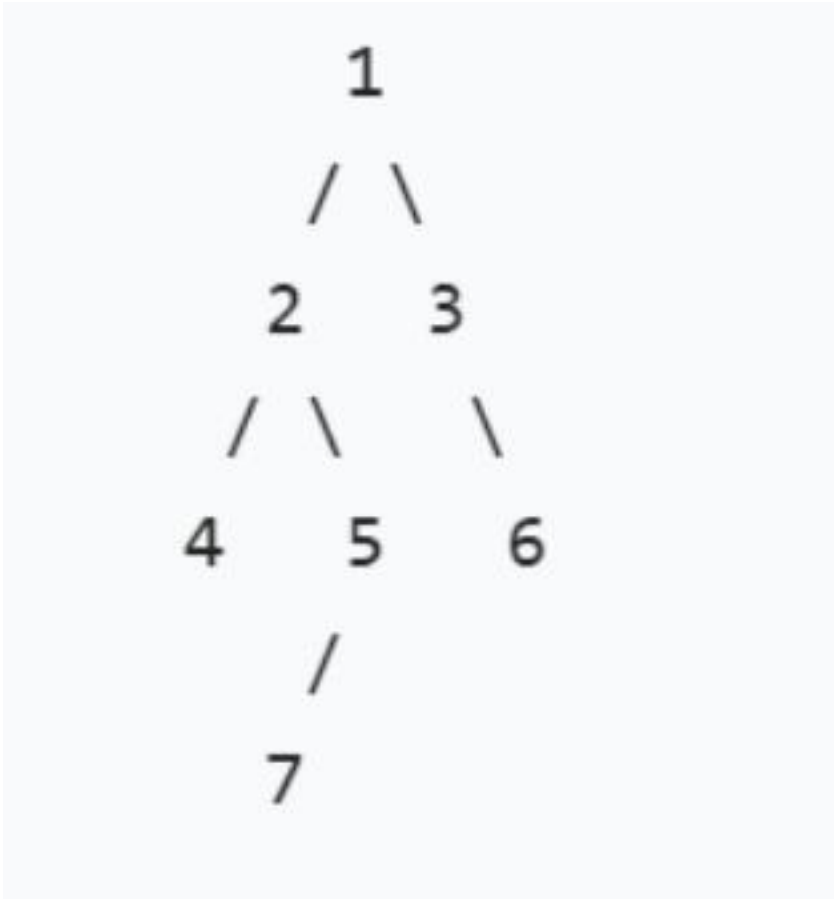
03

遍历

TEACHING
COURSEWARE

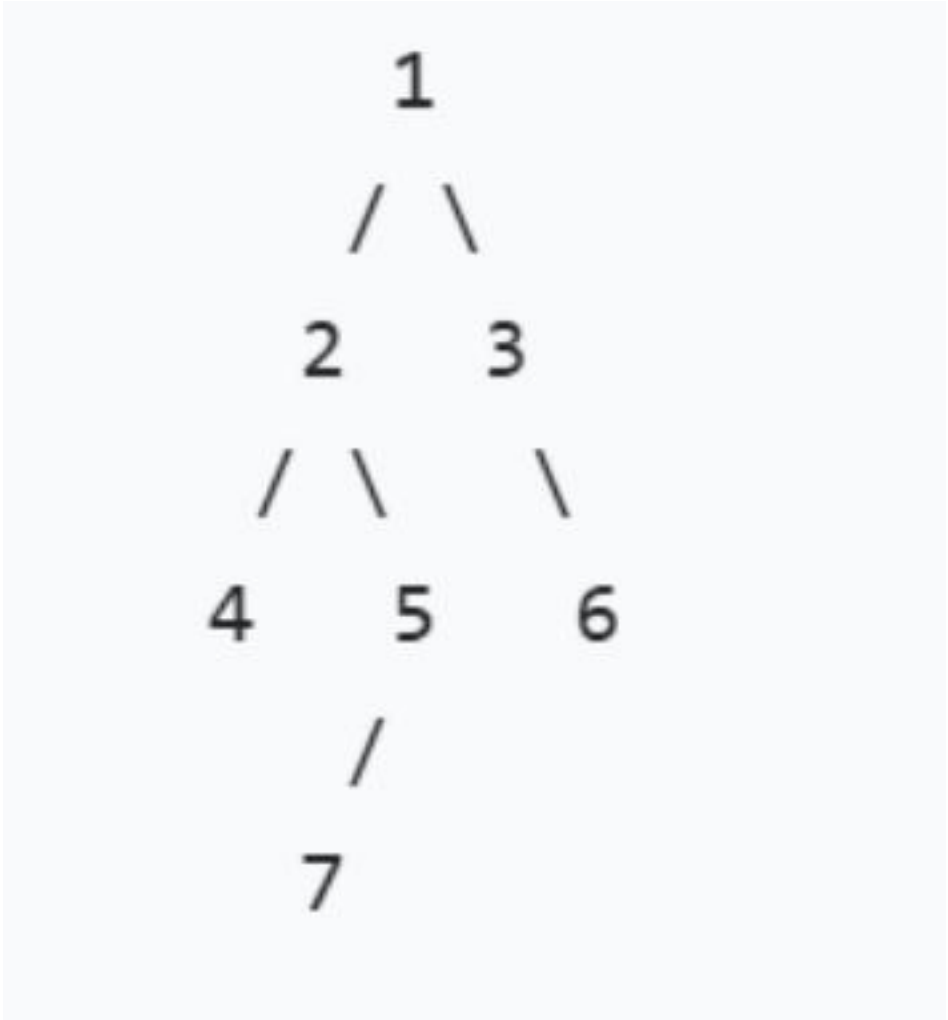
TEACH

遍历



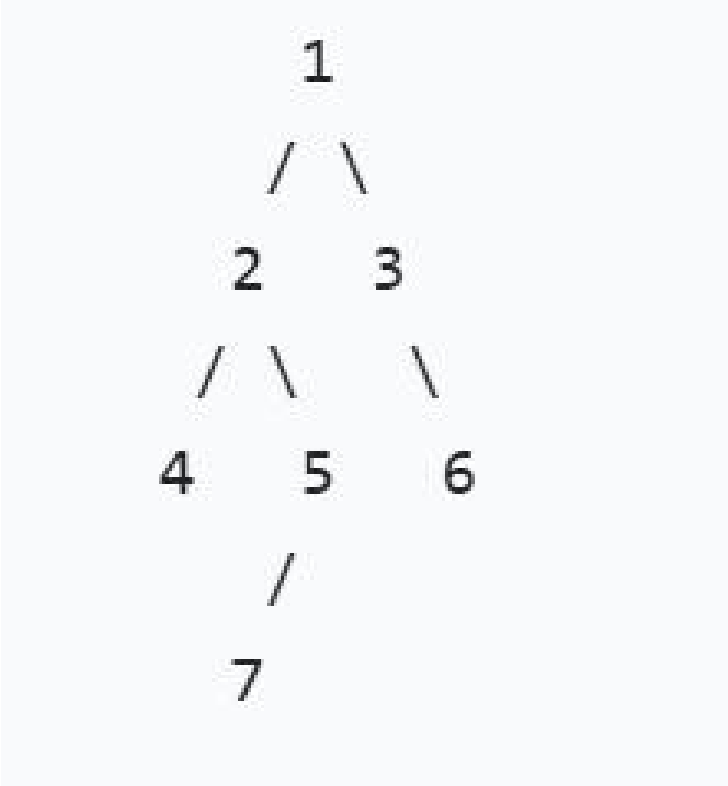
遍历

前序遍历（根→左→右）
遍历逻辑拆解
访问根1；
递归遍历1的左子树（根2）：
访问2；
递归遍历2的左子树（根4）：访问4（无子女，结束）；
递归遍历2的右子树（根5）：
访问5；
递归遍历5的左子树（根7）：访问7（无子女，结束）；
递归遍历5的右子树（空，无操作）；
递归遍历1的右子树（根3）：
访问3；
递归遍历3的左子树（空，无操作）；
递归遍历3的右子树（根6）：访问6（无子女，结束）。
遍历结果
1 → 2 → 4 → 5 → 7 → 3 → 6



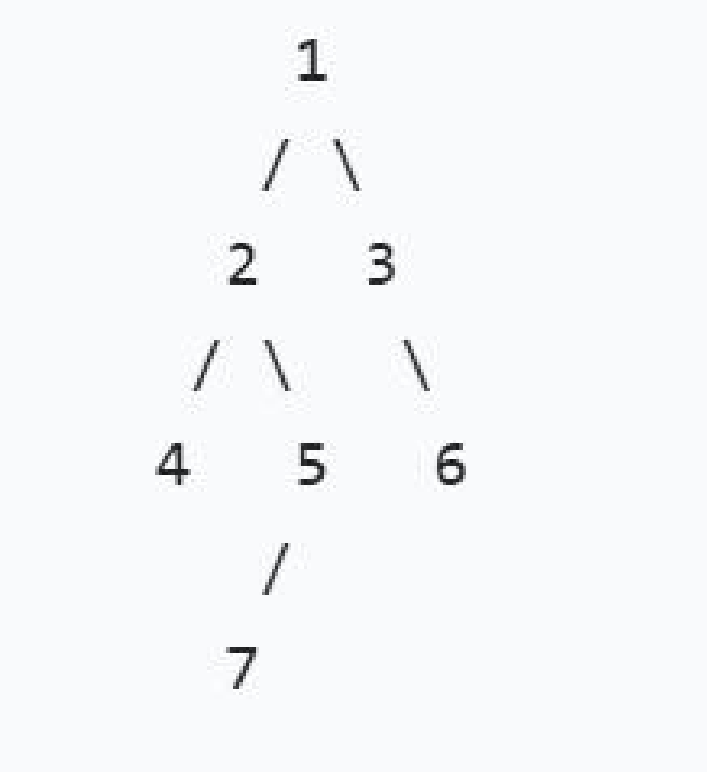
遍历

中序遍历（左→根→右）
遍历逻辑拆解
递归遍历1的左子树（根2）：
递归遍历2的左子树（根4）：访问4（无左子树，结束）；
访问2；
递归遍历2的右子树（根5）：
递归遍历5的左子树（根7）：访问7（无左子树，结束）；
访问5；
递归遍历5的右子树（空，无操作）；
访问根1；
递归遍历1的右子树（根3）：
递归遍历3的左子树（空，无操作）；
访问3；
递归遍历3的右子树（根6）：访问6（无左子树，结束）。
遍历结果
4 → 2 → 7 → 5 → 1 → 3 → 6



遍历

后序遍历（左→右→根）
遍历逻辑拆解
递归遍历1的左子树（根2）：
递归遍历2的左子树（根4）：访问4（无子女，结束）；
递归遍历2的右子树（根5）：
递归遍历5的左子树（根7）：访问7（无子女，结束）；
递归遍历5的右子树（空，无操作）；
访问5；
访问2；
递归遍历1的右子树（根3）：
递归遍历3的左子树（空，无操作）；
递归遍历3的右子树（根6）：访问6（无子女，结束）；
访问3；
访问根1。
遍历结果
4 → 7 → 5 → 2 → 6 → 3 → 1



遍历

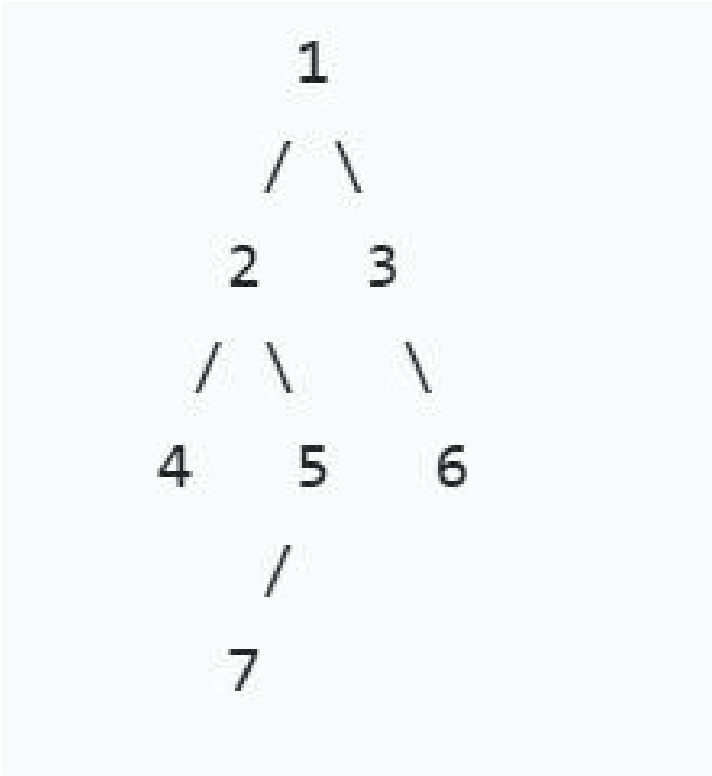
已知前序 + 中序，求后序 为例
前序：1, 2, 4, 5, 7, 3, 6
中序：4, 2, 7, 5, 1, 3, 6

步骤 1：确定整棵树的根与左右子树分割

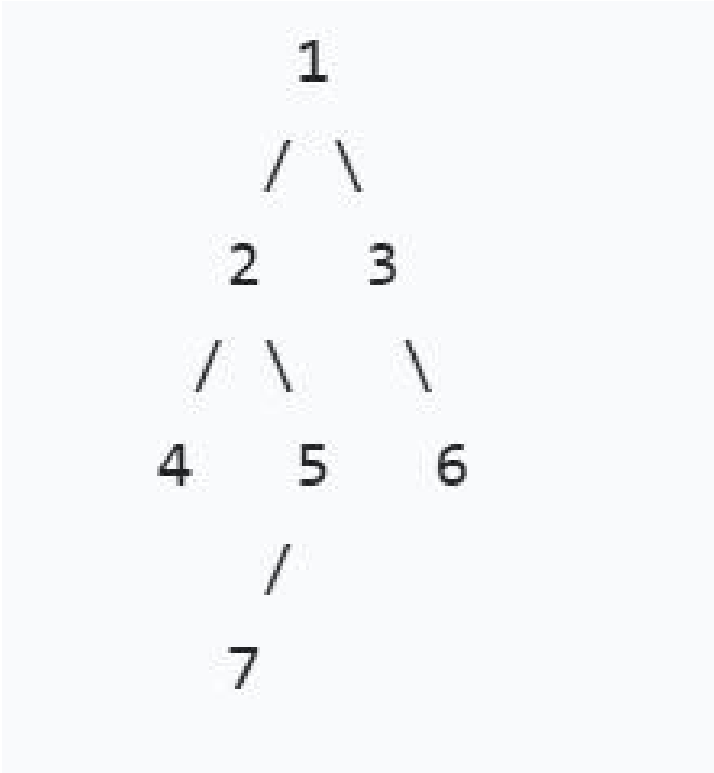
根节点：前序第一个节点1；

中序分割：在中序中找1的位置
左侧[4, 2, 7, 5]为左子树节点
右侧[3, 6]为右子树节点；

前序分割
前序中左子树序列为[2, 4, 5, 7]
右子树序列为[3, 6]



遍历



已知前序 + 中序，求后序 为例
前序：1, 2, 4, 5, 7, 3, 6
中序：4, 2, 7, 5, 1, 3, 6
处理左子树（前序：2, 4, 5, 7；中序：4, 2, 7, 5）

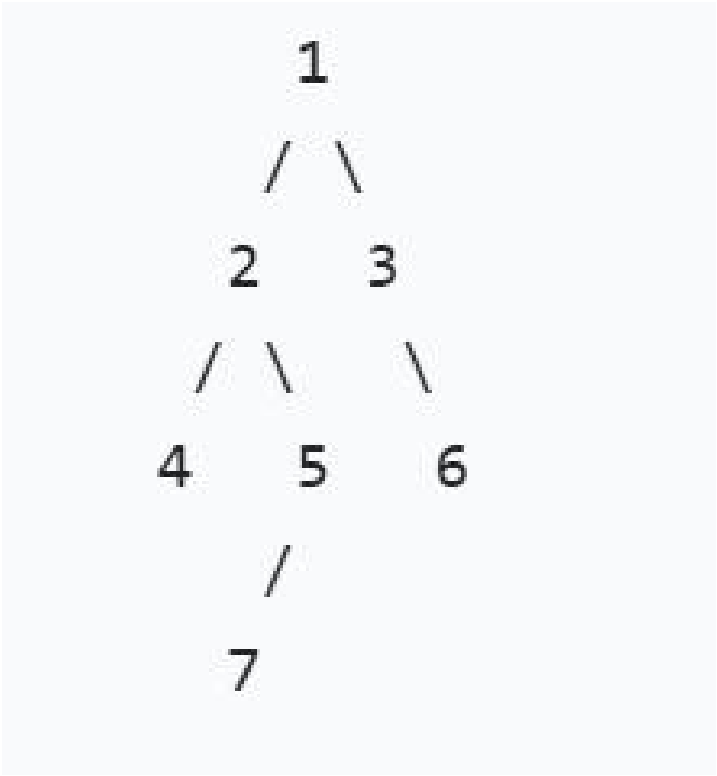
子步骤 2.1：确定左子树的根与分割
根节点：前序第一个节点2；
中序分割：
左侧[4]为左子树节点
右侧[7, 5]为右子树节点；
子步骤 2.2：递归处理2的左子树（前序：4；中序：4）
根为4，无左右子树（中序无分割），后序结果：4。

子步骤 2.3：递归处理2的右子树（前序：5, 7；中序：7, 5）
根节点：前序第一个节点5；
中序分割：左侧[7]为左子树节点，右侧为空；
前序分割：前序左子树序列[7]，右子树为空；
递归处理5的左子树（前序：7；中序：7）：根为7，后序结果7；
此子树后序：7 → 5（左→右→根）。

左子树（根2）的后序结果
4（2的左） → 7→5（2的右） → 2（根），即4, 7, 5, 2。

遍历

已知前序 + 中序，求后序 为例
前序：1, 2, 4, 5, 7, 3, 6
中序：4, 2, 7, 5, 1, 3, 6



步骤 3：递归处理右子树（前序：3, 6；中序：3, 6）

子步骤 3.1：确定右子树的根与分割

根节点：前序第一个节点3；

中序分割：左侧为空，右侧[6]为右子树节点；

前序分割：右子树节点数 = 1，前序右子树序列[6]。

子步骤 3.2：递归处理3的右子树（前序：6；中序：6）

根为6，无左右子树，后序结果：6。

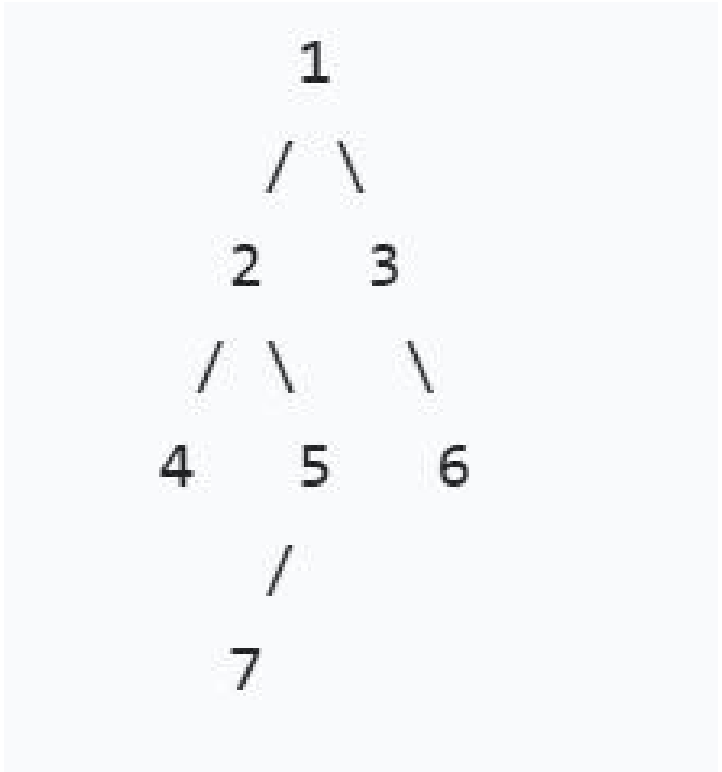
右子树（根3）的后序结果

6（3的右） → 3（根），即6, 3。

合并 4→7→5→2→6→3→1

遍历

已知条件为：
中序：4 → 2 → 7 → 5 → 1 → 3 → 6
后序：4 → 7 → 5 → 2 → 6 → 3 → 1
目标：通过中序 + 后序推导前序遍历结果。



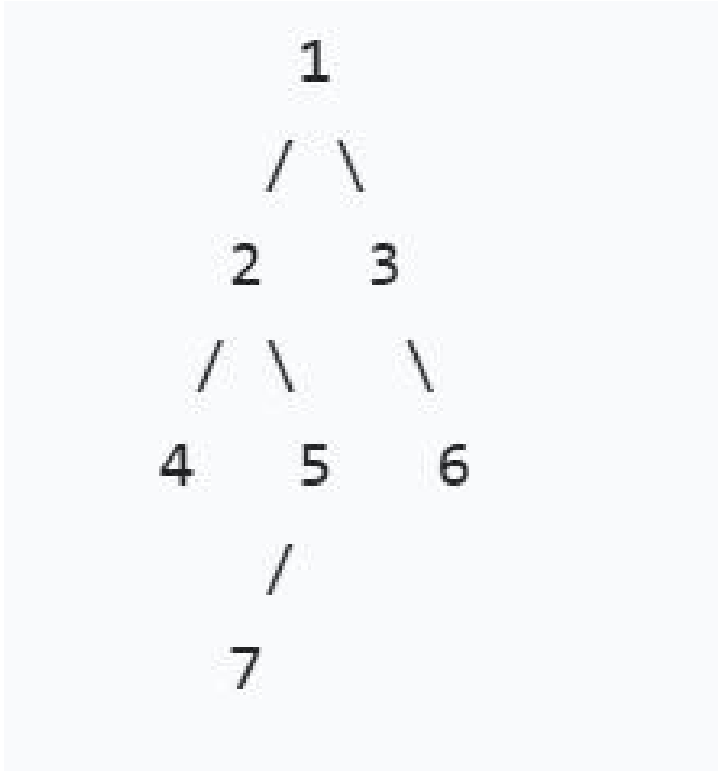
步骤 1：确定整棵树的根与左右子树分割

1. 找根节点：后序遍历最后一个节点为1，因此整棵树的根是1；
2. 中序分割左右子树

左子树节点集（中序左侧）：[4, 2, 7, 5]；
右子树节点集（中序右侧）：[3, 6]；
后序分割左右子树：
左子树后序序列：[4, 7, 5, 2]（后序前 4 个节点）；
右子树后序序列：[6, 3]（后序第 5-6 个节点，排除最后 1 个根1）。

遍历

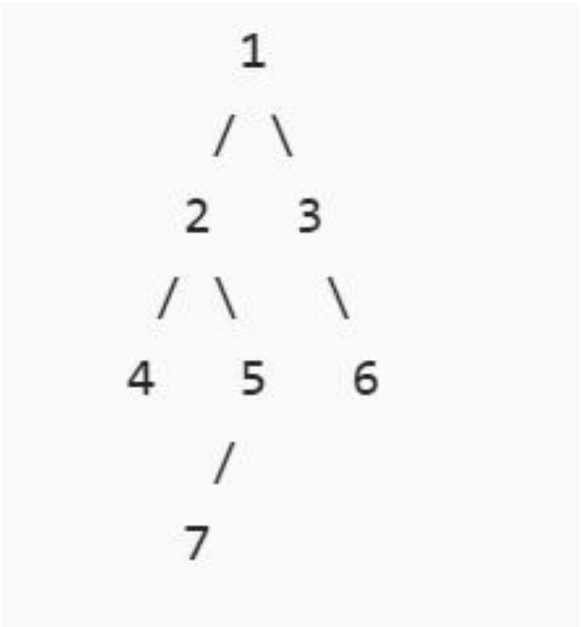
已知条件为：
中序：4 → 2 → 7 → 5 → 1 → 3 → 6
后序：4 → 7 → 5 → 2 → 6 → 3 → 1
目标：通过中序 + 后序推导前序遍历结果。



递归处理左子树（中序：[4, 2, 7, 5]；后序：[4, 7, 5, 2]）
子步骤 2.1：确定左子树的根与分割
找左子树的根：
左子树的左子树节点集（2左侧）：[4]；
左子树的右子树节点集（2右侧）：[7, 5]；
左子树的左子树后序：[4]（前 1 个节点）；
左子树的右子树后序：[7, 5]。
子步骤 2.2：递归处理 “左子树的左子树”（中序：[4]；后序：[4]）
1. 找根：后序最后一个节点为4，此子树无左右子树；
2. 前序片段：仅4。

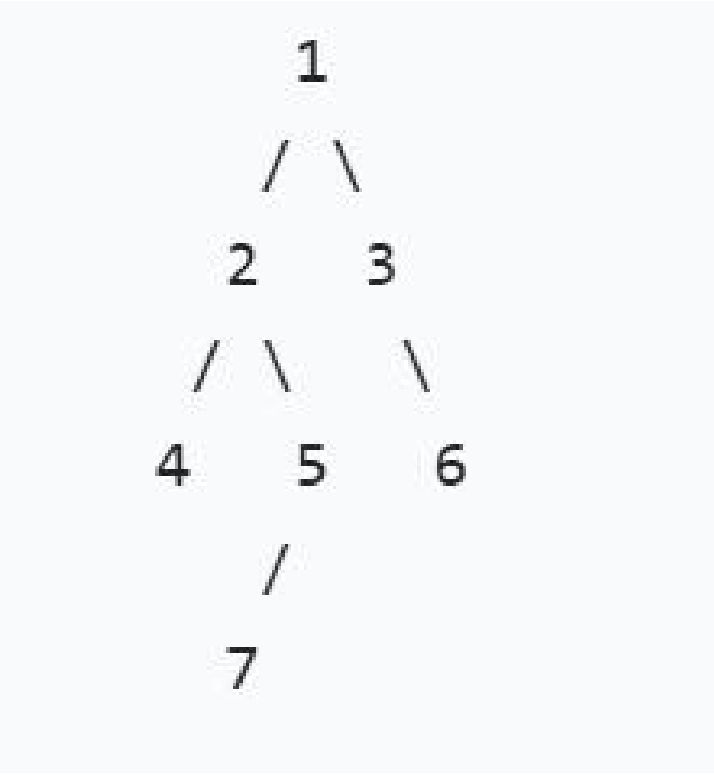
遍历

已知条件为：
中序：4 → 2 → 7 → 5 → 1 → 3 → 6
后序：4 → 7 → 5 → 2 → 6 → 3 → 1
目标：通过中序 + 后序推导前序遍历结果。



子步骤 2.3：处理 “左子树的右子树”
（中序：[7,5]；后序：[7,5]）
找根：后序最后一个节点为5，因此此子树的根是5；
中序分割：在[7,5]中找5的位置：
左子树节点集：[7]（长度 = 1）；
右子树节点集：空（长度 = 0）；
后序分割：
因此左子树后序为[7]
右子树后序为空；
递归处理“5 的左子树”（中序：[7]；后序：[7]）：根为7，前序片段为7；
此子树的前序片段：根5 → 左子树7 → 右子树空，即5→7。
左子树的前序结果
按“根 → 左子树前序 → 右子树前序”
拼接：2（左子树根）→ 4（左子树的左）→ 5→7（左子树的右），最终左子树前序片段为2→4→5→7。

遍历



已知条件为：
中序：4 → 2 → 7 → 5 → 1 → 3 → 6
后序：4 → 7 → 5 → 2 → 6 → 3 → 1
目标：通过中序 + 后序推导前序遍历结果。

步骤 3：递归处理右子树（中序：[3,6]；后序：[6,3]）
子步骤 3.1：确定右子树的根与分割
左子树节点集：空
右子树节点集：[6]
后序分割　因此右子树的右子树后序为[6]。
子步骤 3.2：递归处理 “右子树的右子树”（中序：[6]；后序：[6]）
找根：后序最后一个节点为6，此子树无左右子树；
前序片段：仅6。
右子树的前序结果
按“根 → 左子树前序（空） → 右子树前序”
拼接：3（右子树根）→ 6（右子树的右），最终右子树前序片段为3→6。
步骤 4：合并整棵树的前序结果
前序遍历的规则是“整棵树根 → 左子树前序 → 右子树前序”，因此拼接结果为：
1（整棵树根）→ 2→4→5→7（左子树前序）→ 3→6（右子树前序）
最终前序遍历结果：1 → 2 → 4 → 5 → 7 → 3 → 6，与直接遍历树 S 的前序结果完全一致。

04

题目

TEACHING
COURSEWARE

TEACH

遍历

第 14 题

假设一棵二叉树的后序遍历序列为 DGJHEBIFCA，中序遍历序列为 DBGEHJACIF，则其前序遍历序列为（）。

- A. ABCDEFGHIJ
- B. ABDEGHJCFI
- C. ABDEGJHCFI
- D. ABDEGHJFIC

题目

第 11 题

给定一棵二叉树，其前序遍历结果为：`ABDECFG`，中序遍历结果为：`DEBACFG`。请问这棵树的正确后序遍历结果是什么？

- A. `EDBGFCA`
- B. `EDGBFCA`
- C. `DEBGFCA`
- D. `DBEGFCA`

题目

已知二叉树的前序遍历为 $[A, B, D, E, C, F, G]$，中序遍历为 $[D, B, E, A, F, C, G]$，请问该二叉树的后序遍历结果是？（）

- A. $[D, E, B, F, G, C, A]$
- B. $[D, E, B, F, G, A, C]$
- C. $[D, B, E, F, G, C, A]$
- D. $[D, B, E, F, G, A, C]$